(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平6－223095

(43)公開日　平成６年(1994)８月12日

(51)Int.Cl.[5]		識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ	15/31		Ｍ 7343－5Ｌ		
	11/10	３３０	Ｑ 7313－5Ｂ		
// Ｈ０３Ｍ	13/00		8730－5Ｊ		

審査請求　未請求　請求項の数６　ＯＬ　（全 10 頁）

(21)出願番号　特願平5－9333

(22)出願日　平成５年(1993)１月22日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　岩村　恵市
東京都大田区下丸子３丁目30番２号キヤノン株式会社内

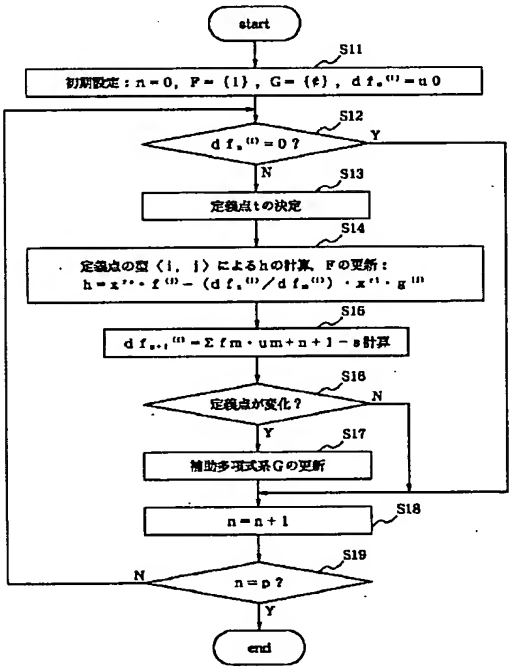
(74)代理人　弁理士　丸島　儀一

(54)【発明の名称】　多項式系導出装置及びその方法

(57)【要約】

【目的】　代数幾何符号の復号において、与えられた多次元配列を生成する最小多項式系を高速に求める。

【構成】　与えられた多次元配列ｕを格納する配列記憶手段と、求める多項式系Ｆ、当該多項式系Ｆと異なる多項式系Ｇをそれぞれ格納するための第１、第２多項式記憶手段と、前記第１、第２多項式記憶手段にそれぞれ記憶された多項式 $f^{(i)}$ ，$g^{(j)}$ に対して、演算 $f^{(k)} = z^{rs} \cdot f^{(i)} - (df_n^{(i)}/df_m^{(j)}) \cdot z^{rt} \cdot g^{(j)}$ を行う第１演算手段と、該演算された多項式 $f^{(k)} = \Sigma f_m^{(k)} \cdot z_m$ と前記記憶された多次元配列ｕに対して演算 $df_{n+1}^{(k)} = \Sigma f_m^{(k)} \cdot u_{m+n+1-s}$ を行なう第２演算手段と、前記第１、及び第２の多項式記憶手段に対するアクセス及び該アクセスのアドレスを多項式 $f^{(k)}$ の次数に依存させて並列に制御する制御手段とを具える。



1

【特許請求の範囲】

【請求項１】　与えられた多次元配列を生成する最小多項式系を求める装置において、

与えられた多次元配列ｕを格納する配列記憶手段と、

求める多項式系Ｆを格納するための第１多項式記憶手段と、

当該多項式系Ｆと異なる多項式系Ｇを格納するための第２多項式記憶手段と、

前記第１多項式記憶手段に記憶された多項式 $f^{(i)}$，前記第２多項式記憶手段に記憶された多項式 $g^{(j)}$ に対して、演算

$$f^{(k)} = z^{rs} \cdot f^{(i)} - (df_n^{(i)}/df_m^{(j)}) \cdot z^{rt} \cdot g^{(j)}$$

を行う第１演算手段と、

該第１演算手段によって演算された多項式 $f^{(k)} = \Sigma f_m^{(k)} \cdot z_m$ と前記配列記憶手段に記憶された多次元配列ｕに対して演算

$$df_{n+1}^{(k)} = \Sigma f_m^{(k)} \cdot u_{m+n+1-s}$$

を行なう第２演算手段と、

前記第１、及び第２の多項式記憶手段に対するアクセス及び該アクセスのアドレスを多項式 $f^{(k)}$ の次数に依存させて並列に制御する制御手段とを有し、前記第１及び第２の演算手段による演算を並列に実行することを特徴とする多項式系導出装置。

【請求項２】　前記第１あるいは第２の多項式記憶手段が、複数のメモリを具え、複数の多項式を、多項式毎に異なるメモリに格納することを特徴とする請求項１に記載の多項式系導出装置。

【請求項３】　前記第１あるいは第２の多項式記憶手段が、複数のメモリを具え、複数の多項式を、変数及び次数毎に異なるメモリに格納することを特徴とする請求項１に記載の多項式系導出装置。

【請求項４】　前記制御手段が、複数の多項式を多項式毎に異なるアドレスに格納するように、前記第１あるいは第２の多項式記憶手段を制御することを特徴とする請求項１に記載の多項式系導出装置。

【請求項５】　前記制御手段が、複数の多項式を変数及び次数毎に異なるアドレスに格納するように、前記第１あるいは第２の多項式記憶手段を制御することを特徴とする請求項１に記載の多項式系導出装置。

【請求項６】　与えられた多次元配列を生成する最小多項式系を求める方法において、

与えられた多次元配列ｕを配列メモリに格納し、

求める多項式系Ｆを格納するための第１多項式メモリに記憶された多項式 $f^{(i)}$，当該多項式系Ｆと異なる多項式系Ｇを格納するための第２多項式メモリに記憶された $g^{(j)}$ に対して、第１の演算

$$f^{(k)} = z^{rs} \cdot f^{(i)} - (df_n^{(i)}/df_m^{(j)}) \cdot z^{rt} \cdot g^{(j)}$$

を実行し、

2

該第１の演算によって求められた多項式 $f^{(k)} = \Sigma f_m^{(k)} \cdot z_m$ と前記配列メモリに記憶された多次元配列ｕに対して第２の演算

$$df_{n+1}^{(k)} = \Sigma f_m^{(k)} \cdot u_{m+n+1-s}$$

を実行し、

前記第１、及び第２の多項式メモリに対するアクセス及び該アクセスのアドレスを多項式 $f^{(k)}$ の次数に依存させて並列に制御し、前記第１及び第２の演算を並列に実行することを特徴とする多項式系導出方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、ディジタル通信系及びディジタル記憶系において通信路または記憶媒体で受けた誤りを、誤り訂正符号を用いて受信側で自動的に訂正する誤り訂正に関し、特に誤り訂正符号として代数幾何符号を用いる場合の復号において、与えられた多次元配列を生成する最小多項式系を求める多項式系導出装置及びその方法に関する。

【０００２】

【従来の技術】従来、ディジタル通信系及びディジタル記憶系において通信路または記憶媒体で受けた誤りを、受信側で自動的に訂正する誤り訂正符号として、リード・ソロモン（ＲＳ）符号[1]やＢＣＨ符号[1]がよく知られ、コンパクト・ディスクや衛星通信等において実際に使用されている。そのＲＳ符号やＢＣＨ符号の復号法として、バーレカンプ・マッセイ（ＢＭ）法がよく知られ、装置化されている。また、最近は代数曲線論を駆使した代数幾何符号[1]-[4]と呼ばれる符号の研究が盛んに行われている。代数幾何符号は前述のＲＳ符号やＢＣＨ符号をその１クラスとして含む非常に適用範囲の広い符号系であり、従来の符号よりもよい性質を持つ新符号が存在することが徐々に明らかになってきている[5][6]。その復号法として阪田によって提案されたアルゴリズムが知られている[7]。このアルゴリズムは２次元ＢＭ法と呼ばれ、ＲＳ符号の復号に用いられるＢＭ法（以後、１次元ＢＭ法と呼ぶ）の拡張になっていることが知られている。また、ＢＭ法は阪田によって多次元にまで拡張され[8]、図７のような構造を持つアルゴリズムとして表される。（ただし、定義点の決定法、及びｒｓ，ｒｔ等は文献[7][8]参照。）

【０００３】

【発明が解決しようとする課題】図７のｎは全順序（文献[7][8]参照）と呼ばれる順序づけによって更新される。図７において２）の $df_n^{(i)}$ の計算と４）のｈによる $f_n^{(i)}$ の更新は逐次的に行われることが多かった。なぜならば、点ｎにおける２）の $df_n^{(i)}$ は点ｎ－１における４）のｈを用いて計算され、点ｎにおける４）のｈによる $f_n^{(i)}$ の更新は点ｎの２）の $df_n^{(i)}$ を用いて計算されるためである。この場合、図８に示すように処理時間に無駄が生じるために効率的ではなかった。

3

【０００４】一方、ＲＳ符号やＢＣＨ符号の復号法である１次元ＢＭ法の１次元変数についての処理を逐次的に行うことを利用して２）と４）の処理を並列に実行する手法が文献[9]に提案されている。しかし、この手法は段数固定のシフトレジスタを用いているために１回の$f_n^{(i)}$の更新（１サイクル）に必ずｔ＋１（ｔは多項式$f_n^{(i)}$の最終点（ｎ＝ｐ：図７参照）における定義点の最大値）の処理クロックが必要であった。従って、文献[9]の装置は最大の定義点とならない$f_n^{(i)}$の途中演算（ｎ＜ｐ）において無駄な処理クロックが必要であり、更に多次元のＢＭ法を考慮していないために、次の点において文献[9]の手法は多次元ＢＭ法には不向きであった。

【０００５】１）　１次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式は１つであるが、多次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式は複数存在する。

【０００６】２）　１次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式の次数と与えられる配列の配置は１次元（１変数）であるので、高次から順に１次元状のメモリ（シフトレジスタ等）に各係数を格納できるが、多次元ＢＭ法の多項式の次数と配列の配置は１次元ではないので１次元状のメモリでは効率的な処理ができない。

【０００７】３）　１次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式は１つであるので、多項式間の並列性は意味が無いが、多次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式は複数存在するので、多項式間の演算も並列化できる。

【０００８】４）　１次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式の変数は１つであるので、１つの変数についての演算を並列に行うと２）と４）の処理は並列化できないが、多次元ＢＭ法では多項式系Ｆ，Ｇに属す多項式の変数は複数存在するので、１つの変数についての演算を並列に行っても２）と４）の処理を並列化できる。

【０００９】そこで、本発明では上述の欠点を除去し、多次元ＢＭ法に対しても２）と４）の処理を効率的に並列処理するアルゴリズム及び、それを実現する装置を提案することを目的とする。これによって、１次元ＢＭ法を含む多次元ＢＭ法の処理時間が短縮され、高速な処理が実現できる。

【００１０】［参考文献］

[1] 今井：“符号理論”，電子情報通信学会

[2]V.D.Goppa: “Codes on algebraic curves”, Soviet Math.Dokl., 24, pp.170-172, 1981.

[3]V.D.Goppa: “Algebraico-geometric codes”, Math. U.S.S.R. Izvestiya,vol.21, no.1, pp.75-91, 1983.

[4]V.D.Goppa: “Geometry and Codes”, Kluwer Academic Publishers, 1988.

[5]M.A.Tsfasman and S.G.Vladut: “Algebraic-geometric codes”, KluwerAcademic Publishers, 1991.

4

[6] 三浦：“ある平面曲線上の代数曲線符号”，Preprint.

[7] 阪田：“与えられた２次元配列を生成する２次元線形帰還シフトレジスタの合成”，信学論（Ａ），vol.J-70A, pp.903-910, 1987.

[8]S.Sakata: “Extension of the Berlekamp-Massey algorithm to N dimensions,” Information and Computation, vol.84, pp.207-239, 1990

[9]Youzhi Xu: “Contributions to the Decoding of Reed-Solomon and Related Codes,” Linkoping Studies in Science and Technology. Dissertations No.257. 1991.

【００１１】

【課題を解決するための手段】上記課題を解決するために、本発明では、与えられた多次元配列を生成する最小多項式系を求める装置に、与えられた多次元配列ｕを格納する配列記憶手段と、求める多項式系Ｆを格納するための第１多項式記憶手段と、当該多項式系Ｆと異なる多項式系Ｇを格納するための第２多項式記憶手段と、前記第１多項式記憶手段に記憶された多項式$f^{(i)}$，前記第２多項式記憶手段に記憶された多項式$g^{(j)}$に対して、演算$f^{(k)}=z^{r_0}\cdot f^{(i)}-(df_n^{(i)}/df_m^{(j)})\cdot z^{r_t}\cdot g^{(j)}$を行う第１演算手段と、該第１演算手段によって演算された多項式$f^{(k)}=\Sigma f_m^{(k)}\cdot z_m$と前記配列記憶手段に記憶された多次元配列ｕに対して演算$df_{n+1}^{(k)}=\Sigma f_m^{(k)}\cdot u_{m+n+1-s}$を行なう第２演算手段と、

前記第１、及び第２の多項式記憶手段に対するアクセス及び該アクセスのアドレスを多項式$f^{(k)}$の次数に依存させて並列に制御する制御手段とを具える。

【００１２】

【作用】与えられた多次元配列ｕを配列記憶手段に格納し、求める多項式系Ｆを第１多項式記憶手段に、当該多項式系Ｆと異なる多項式系Ｇを第２多項式記憶手段にそれぞれ格納するようにし、第１演算手段によって前記第１多項式記憶手段に記憶された多項式$f^{(i)}$，前記第２多項式記憶手段に記憶された多項式$g^{(j)}$に対して、演算$f^{(k)}=z^{r_0}\cdot f^{(i)}-(df_n^{(i)}/df_m^{(j)})\cdot z^{r_t}\cdot g^{(j)}$を実行し、該第１演算手段によって演算された多項式$f^{(k)}=\Sigma f_m^{(k)}\cdot z_m$と前記配列記憶手段に記憶された多次元配列ｕに対して、第２演算手段によって演算$df_{n+1}^{(k)}=\Sigma f_m^{(k)}\cdot u_{m+n+1-s}$を実行し、制御手段が前記第１、及び第２の多項式記憶手段に対するアクセス及び該アクセスのアドレスを多項式$f^{(k)}$の次数に依存させて並列に制御することで、前記第１及び第２の演算手段による演算を並列に実行する。

【００１３】

【実施例】

〔説明の概要〕Ｎ次元ＢＭ法において２点間の加減算は各次元の変数について独立に行われる。そのために、Ｎ次元ＢＭ法においてはＮ次元の変数を独立に扱うＮ次元

5

状のメモリを用いた方が効率的な処理を行うことができる。

【００１４】更に、Ｎ－１次元の変数についての演算を同時に行っても、残りの変数についての処理が逐次的に行われるので、２）と４）の処理を並列化できる。Ｎ－１次元の変数についての演算を同時に行うためには、その変数について各次数毎に独立したメモリを用いて同時にアクセスすればよい。従って、本発明では多項式及び各変数の次数毎に独立のメモリをメモリのアドレス分割または複数のメモリによって実現し、効率的に多次元ＢＭ法の処理を行うアルゴリズム及び装置を提案する。

【００１５】図７に対応する本発明によるアルゴリズムを図１に、それによる動作を図２に示す（図２(a) ～(c) は実施例１～３に対応する）。図１において２）と４）及び５）の処理は並列に行われている。

【００１６】〔実施例１〕まず１次元ＢＭ法について考える。図３においてｕは与えられた１次元配列ｕ（アドレスｉにはｕｉ が設定）を格納するメモリ、ｆはｕを生成する最小多項式ｆ（初期値は１）を格納するメモリ、ｇはｆの補助多項式ｇ（初期値は０）を格納するメモリであり、そのアドレスは外部から制御される。ここでは１次元ＢＭ法を考えているのでメモリに格納される多項式は１つであり、その多項式の最高次数をｓとした場合ｓ－ｉ次の次数の係数はメモリのアドレスｉに設定される。また、df_n は$df_n^{(i)}$ を計算するためのレジスタ、df_m は前のdf_n の値を保持しておくレジスタ、INは逆数を出力する回路であり、ddは（$df_n^{(i)}$ ／$df_m^{(j)}$）の値を保持するレジスタ（初期値はｕ０）である。さらに、SWは定義点は変化した場合のみメモリｆからの出力を選択するスイッチであり、その制御は外部から行われる。また、×，＋は各々乗算器，加算器を表す。

【００１７】点ｎにおいて、多項式ｆの定義点を$s^{(i)}$、多項式ｇの定義点を$s^{(j)}$とする。メモリｆ，ｇは各々アドレス０から１ずつ順次加算され、アドレス$s^{(i)}$，$s^{(j)}$まで制御される。このとき、外部回路は定義点から計算されるｒｓ＋$s^{(i)}$とｒｔ＋$s^{(j)}$の大小関係を比較し、大きい方のメモリから動作させはじめ、その差分のクロック数後には他方のメモリも動作させる。前述したようにddのレジスタは（$f_n^{(i)}$ ／$df_m^{(j)}$）の値を保持しているので、メモリｆには４）のｈによって演算・更新された点ｎにおける多項式ｆの各係数がアドレス０から$s^{(k)}$（$s^{(k)}$は更新された多項式の定義点）へ順次入力される。このとき、更新された多項式ｆの各係数に合わせて、メモリｕはアドレスｎ＋１から順次１ずつ減じてアドレスｎ＋１－$s^{(k)}$まで制御され、そこに蓄えている１次元配列ｕの各要素を順次出力する。これによって、２）の点ｎ＋１における$df_{n+1}^{(k)}$が４）の点ｎにおけるｆと並列に計算される。このときレジスタdf_mには前の$df_n^{(i)}$が入力され、INによるその逆数と$df_{n+1}^{(k)}$との乗算を行うことにより、次の演

6

算におけるddをレジスタに保持することができる。ただし、定義点が変化した場合にはSWが開き、メモリｇにはメモリｆの出力がアドレス０からアドレス$s^{(i)}$に対して順次入力され、多項式ｇが更新される。従って、図３において多項式ｆの１回の更新（１サイクル）には$s^{(k)}$のクロック数が必要で、これをｐ回繰り返すことによりｕを生成する最小多項式ｆを得ることができる。

【００１８】$s^{(k)}$は$df_{n+1}^{(k)}$及び多項式ｆの各係数を逐次処理する場合の最小クロック数である。従って、本発明は文献[9] の装置に比べて無駄な処理クロックがなくなり処理が高速化される。また、定義点の決定及び比較は高速性を要求されないので、通常のＣＰＵによって実現できる。従って、外部回路はＣＰＵとアドレス制御を行うカウンタ回路によって簡単に実現できる。

【００１９】〔実施例２〕実施例１に示したメモリのアドレス制御を行う外部回路は１次元ＢＭ法に対しては無駄な処理クロックを省くだけであったが、２次元以上のＢＭ法に対しては前述の多次元状のメモリをアドレス制御によって実現し、効率的な多次元ＢＭ法を実行するために重要な意味を持つ。

【００２０】まず簡単のために、２次元ＢＭ法について考える。図４においてＵは与えられた多次元配列ｕを、Ｆはｕを生成する複数の多項式$f^{(i)}$ (i=0, …, I-1) を、ＧはＦの複数の補助多項式$g^{(j)}$ (j=0, …, I-2) を格納するメモリである。$df_n^{(i)}$は$df_n^{(i)}$ を計算するためのレジスタ、$df_m^{(j)}$は前の$df_n^{(i)}$の値を保持しておくレジスタであるが、多項式が複数あるのでこのレジスタも複数必要である。他の部品は実施例１と同じである。

【００２１】ここで、ｉを多項式$f^{(i)}$の数，ｊをy^jの次数，ｋを$x^{s^{(i)}-k}$の次数（$s^{(i)}$は多項式$f^{(i)}$の定義点）とすると、メモリＦ，Ｇは図５に示すようなアドレス分割が行え、そのアドレスは（ｉ，ｊ，ｋ）と表現できる。ただし、図５におけるｓｊはy^jに対するｘの最大次数を表す。このとき、多項式系Ｆの中の１つの多項式$f^{(i)}$の定義点を$s^{(i)}$＝（$s1^{(i)}$，$s2^{(i)}$）とし、ａ＝（ａ１，ａ２）（$a \in \Sigma_0^{s^{(i)}}$：文献[7][8]参照）を変数とすると、多項式$f^{(i)}$はアドレス（ｉ，$s1^{(i)}$－ａ１，ａ２）の位置に格納されている。また、多項式系Ｇの中の１つの多項式$g^{(j)}$の定義点を$s^{(j)}$＝（$s1^{(j)}$，$s2^{(j)}$）とすると、多項式ｇ(j) は$a \in \Sigma_0^{s^{(j)}}$に対してアドレス（ｊ，$s1^{(j)}$－ａ１，ａ２）に格納されている。この場合、メモリＦ，Ｇに用いられる各ａは全順序に従う必要はなく、ａ１とａ２と独立に制御することができる。例えば、最初ａ１＝ａ２＝０としてａ１を０から$s0$まで動作させた後、ａ２を１繰上げ、ａ１を再び０から$s1$まで動作させるような制御を行ってもよい。従って、ａ１ａ２は通常のアップカウンタによってに簡単に制御できる。また、メモリＵは点ｎ＝（ｎ１，ｎ２）の写像ｕｎをアドレス（ｎ１，ｎ２）に割り付ければ、（ｊ，ｋ）と表されるアドレス分割が

できる、メモリF，Gと同様に簡単なアドレス制御が行える。

【0022】従って、図4の装置は次のように動作させればよい。まず、外部回路は定義点から計算されるrs+$s^{(i)}$とrt+$s^{(j)}$の全順序による大小関係を比較し、大きい方のメモリから動作させその多項式を出力する。その差分のクロック数後に他方のメモリも動作させ他方の多項式を出力する。ddのレジスタには（$df_n^{(i)}$/$df_m^{(j)}$）の値が保持されているので、メモリFには4）のhによって演算・更新された定義点$s^{(k)}$＝（$s1^{(k)}$，$s2^{(k)}$）を持つ多項式$f^{(k)}$がアドレス（k，s1(k)－a1，a2）（$a \in \Sigma_0^{s^{(k)}}$）に対して順次入力されることがわかる。このとき、点nの全順序における次の点をn＋1＝n'＝（n1'，n2'）とすると、更新された多項式$f^{(k)}$のアドレス内の変数aに合わせて、メモリUはアドレス（n1'－a1，n2'－a2）（$a \in \Sigma_0^{s^{(k)}}$）に蓄えている1次元配列uの各要素を出力する。これによって、2）の点n＋1における$df_{n+1}^{(k)}$が4）の点nにおける多項式$f^{(k)}$と並列に計算される。このときレジスタ$df_n^{(i)}$には2）で計算された$df_{n+1}^{(k)}$が、レジスタdf_mには前の$df_n^{(i)}$が蓄えられ、後の演算において必要なときに用いられる。よって、次の演算において必要な（$df_n^{(i)}$／$df_m^{(j)}$）の値がddのレジスタに保持され図1のアルゴリズムが連続して実行されていく。ただし、定義点が変化した場合にはSWが開き、メモリGにはメモリFのそのときの出力がアドレス（i，$s1^{(i)}$－a1，a2）（$a \in \Sigma_0^{s(i)}$）に対して順次入力され、多項式系Gが更新される。

【0023】従って、図4において1つの多項式$f^{(i)}$の更新（1サイクル）には$s^{(k)}$クロックが必要である。多次元BM法ではステップS14で計算される多項式$f^{(k)}$とそれに用いられる多項式$f^{(i)}$，$g^{(j)}$のk，i，jは定義点の型によって決定される。前述したように複数の多項式をアドレス分割して格納した場合、そのk，i，jはアドレスの指定によって順次選択できる。従って、点nにおいて指定された全ての多項式$f^{(k)}$を更新するためには、各$s^{(k)}$の和に相当するクロック数が必要になる。さらに、それを点pまで繰り返せば2次元配列uを生成する最小多項式系Fが得られる。

【0024】この多項式の選択も定義点を決定する外部回路によって実行できることは明かである。従って、アドレスを制御する外部回路を用いて複数の多項式をアドレス分割により選択することによって従来の文献[9]の装置で困難であった1）の問題を解決できる。さらに、問題2）に示された多次元の変数に関するメモリアクセスの複雑さの問題も図5に示すようにyの次数毎にアドレス分割を行うことによって容易になり解決できる。

【0025】また、各多項式は最小のクロック数$s^{(k)}$によって更新されるので、実施例1と同様に処理クロックを無駄にしない効率的な処理が行われていることは明

かである。

【0026】また、図4の回路がアドレスの分割数さえ増せば、N次元BM法に対しても有効であることも明かである。

【0027】〔実施例3〕実施例2は複数の多項式を1つのメモリのアドレス分割によって格納しているので、多項式及び各次数は1つしか選択できず問題3），4）に示す多項式間及び変数毎の並列処理はなされていない。そこで、実施例3では問題3），4）に示した多項式間及び変数毎の並列性を実現する装置を考える。

【0028】まず、多項式間の並列性を実現する本発明による実施例を図6に示す。図6においてメモリF，Gは複数の多項式を並列に格納する複数のメモリによって構成される。図6において太線で示される線は複数のメモリ数に応じた多重の信号線を意味し、＋，×は多重の信号線に応じて並列に並べた複数の加算器及び乗算器を示す。また、レジスタ$df_n^{(i)}$，$df_m^{(j)}$，dd及び逆数生成回路INも多重の信号線に応じて並列に並べられる。ただし、メモリUに格納されているuだけは1種類であり、複数の多項式$f^{(i)}$に対して共通であるので1本の線で表され、並列に並べられた乗算器に共通に入力される。

【0029】このとき、メモリF，Gのアドレスは各点nにおける最大の定義点を持つ多項式について制御すれば、他の多項式もそのアドレス制御によって制御できる。従って、点nにおける全ての多項式の更新にはその点における最大定義点である$s^{(k)}$のクロック数でよく、それをp回繰り返すことによってuを生成する最小多項式系Fが得られる。

【0030】次に、変数毎の並列性を実現する装置も図6の実施例によって実現できる。この場合、実施例2の多項式のyの次数毎のアドレス分割を複数のメモリによって並列に格納すれば、1つの多項式内における変数yについての演算を並列に実現することができるので、多項式間だけでなく1つの多項式内の変数毎の並列処理を実現することができる。

【0031】また、図6の回路はメモリ数さえ増せば、N次元BM法に対しても有効であることも明らかである。

【0032】〔その他の実施例〕実施例3の装置のメモリ数は多項式分または次数分の数でなくても実施例2の場合と組み合わせて、準備できる複数のメモリをさらにアドレス分割して用いることもできる。

【0033】また、2）及び4）の処理を行う回路はCPU等によっても容易に構成することができ、本実施例に示した回路構成に制限されない。

【0034】また、ここに示したメモリの格納法及びアクセス法等は、多次元BM法に限らず、他の多次元配列及び多次元多項式生成法に関しても有効であることはいうまでもない。

【0035】

9

【発明の効果】本発明は１サイクルを逐次処理に必要な最小のクロック数 $s^{(k)}$ で実行できるので、無駄な処理クロックがなくなり処理が高速化される。

【００３６】また、複数の多項式とその変数をアドレス分割を用いて制御することによって、多次元ＢＭ法にも簡単に対応することができる。

【００３７】さらに、多次元ＢＭ法の複数の多項式を複数のメモリに分割して並列に制御することによって、複数の多項式を並列に処理することができ処理速度を向上させることができる。

【００３８】さらに、多次元ＢＭ法の多項式内の変数の次数毎のアドレス分割を複数のメモリによって並列に格納することによって、１つの多項式内の変数毎の処理を並列に処理することができ処理速度をより向上させることができる。

【図面の簡単な説明】

【図１】本発明による多次元ＢＭ法のアルゴリズムである。

10

【図２】図１のアルゴリズムを実行した場合の動作図である。

【図３】図１のアルゴリズムの第１の実施例である。

【図４】図１のアルゴリズムの第２の実施例である。

【図５】メモリのアドレス分割の説明図である。

【図６】図１のアルゴリズムの第３の実施例である。

【図７】従来の多次元ＢＭ法のアルゴリズムである。

【図８】図７のアルゴリズムを実行した場合の動作図である。

【符号の説明】

＋　加算器

×　乗算器

ＳＷ　スイッチ

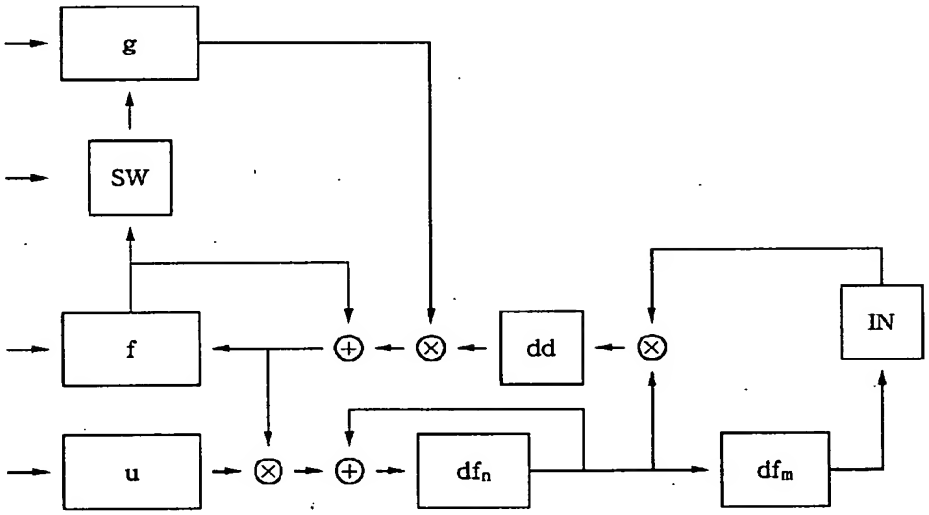
ｕ，Ｕ　与えられた配列を格納するメモリ

ｆ，ｇ，Ｆ，Ｇ　多項式を格納するメモリ

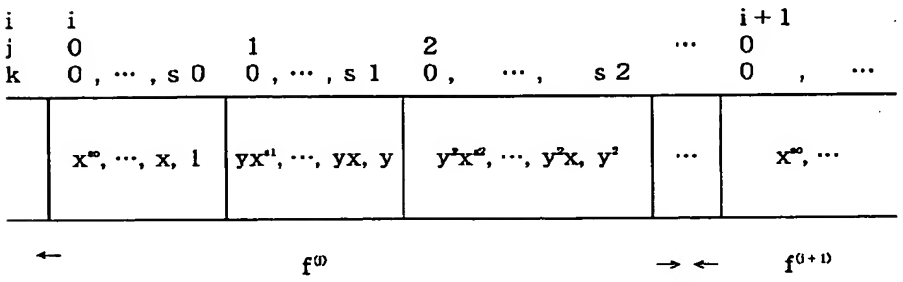
ｄｄ，df_n，df_m，$df_n^{(i)}$，$df_m^{(j)}$　レジスタ

ＩＮ　逆数を出力する回路

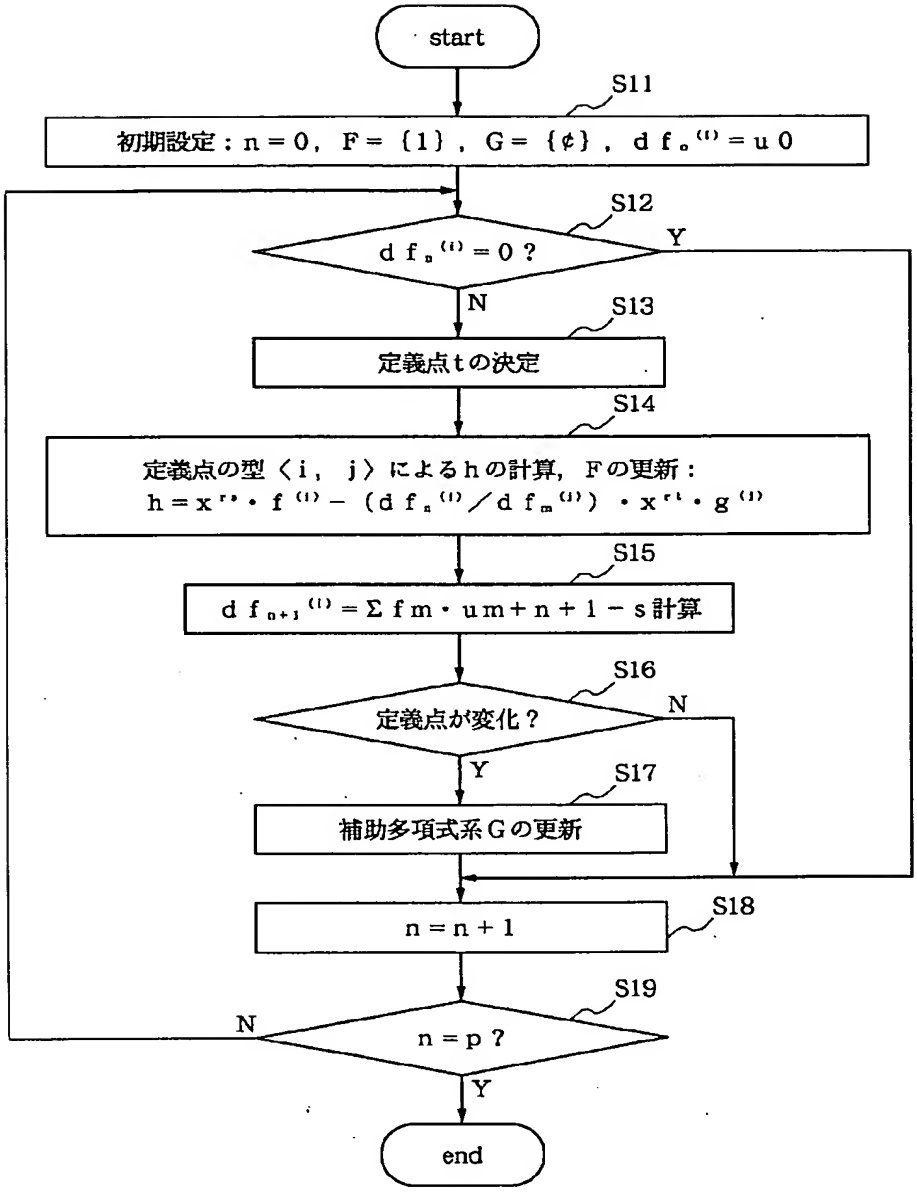
【図３】



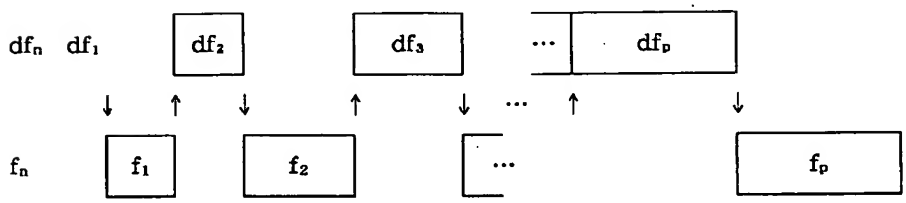
【図５】



【図１】

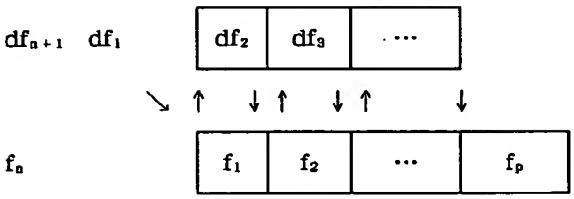

start
S11
初期設定：n＝0，F＝｛1｝，G＝｛¢｝，df₀⁽ⁱ⁾＝u0
S12
df₀⁽ⁱ⁾＝0？
Y
N
S13
定義点tの決定
S14
定義点の型〈i，j〉によるhの計算，Fの更新：
h＝x^r・f⁽ⁱ⁾－（df₀⁽ⁱ⁾／df_m⁽ʲ⁾）・x^t・g⁽ʲ⁾
S15
df_{n+1}⁽ⁱ⁾＝Σfm・um+n+1－s計算
S16
定義点が変化？
N
Y
S17
補助多項式系Gの更新
n＝n＋1
S18
S19
n＝p？
N
Y
end

【図８】

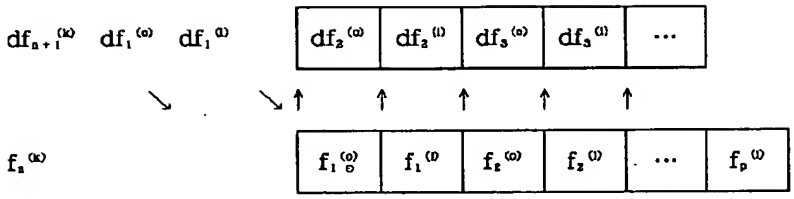

dfn
df1
df2
df3
…
dfp
fn
f1
f2
…
fp

【図２】

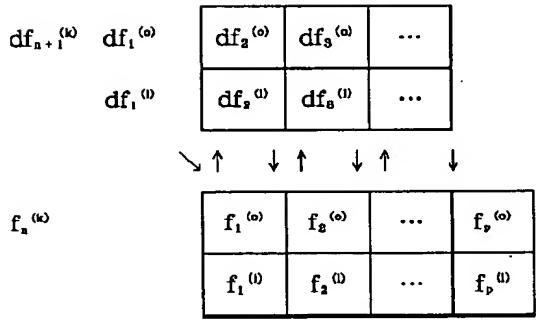
(a)


df_{n+1} df_1
df_2 df_3 ・・・
f_n
f_1 f_2 ・・・ f_p

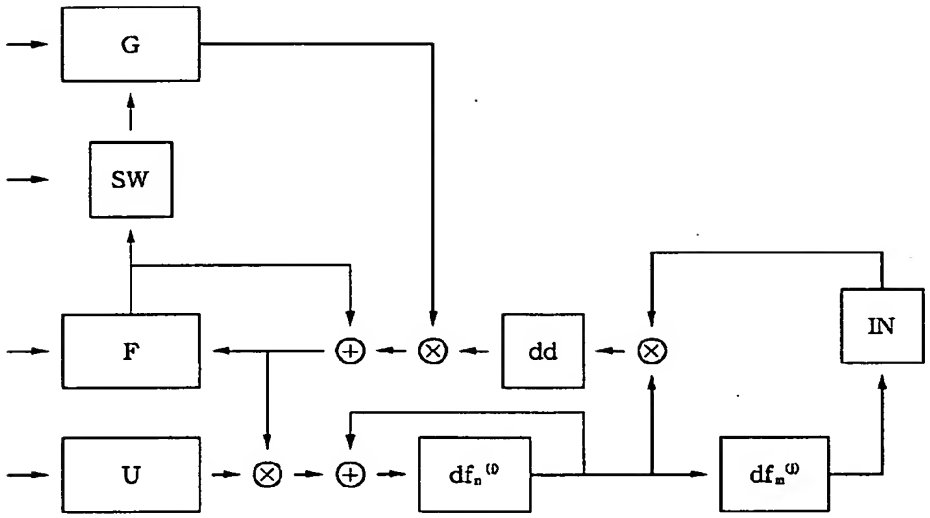
(b)


$df_{n+1}^{(k)}$ $df_1^{(o)}$ $df_1^{(1)}$
$df_2^{(o)}$ $df_2^{(1)}$ $df_3^{(o)}$ $df_3^{(1)}$ ・・・
$f_n^{(k)}$
$f_1^{(o)}$ $f_1^{(1)}$ $f_2^{(o)}$ $f_2^{(1)}$ ・・・ $f_p^{(1)}$

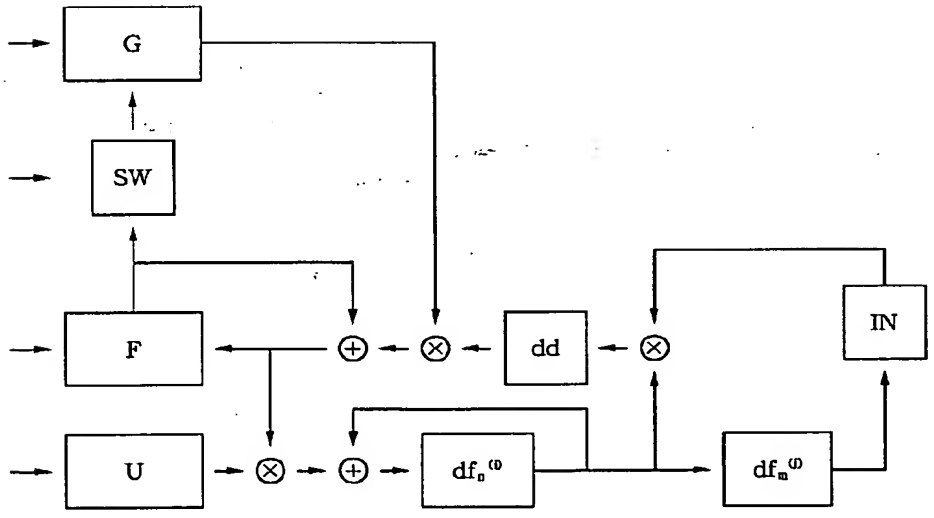
(c)


$df_{n+1}^{(k)}$ $df_1^{(o)}$
$df_2^{(o)}$ $df_3^{(o)}$ ・・・
$df_1^{(1)}$
$df_2^{(1)}$ $df_3^{(1)}$ ・・・
$f_n^{(k)}$
$f_1^{(o)}$ $f_2^{(o)}$ ・・・ $f_p^{(o)}$
$f_1^{(1)}$ $f_2^{(1)}$ ・・・ $f_p^{(1)}$

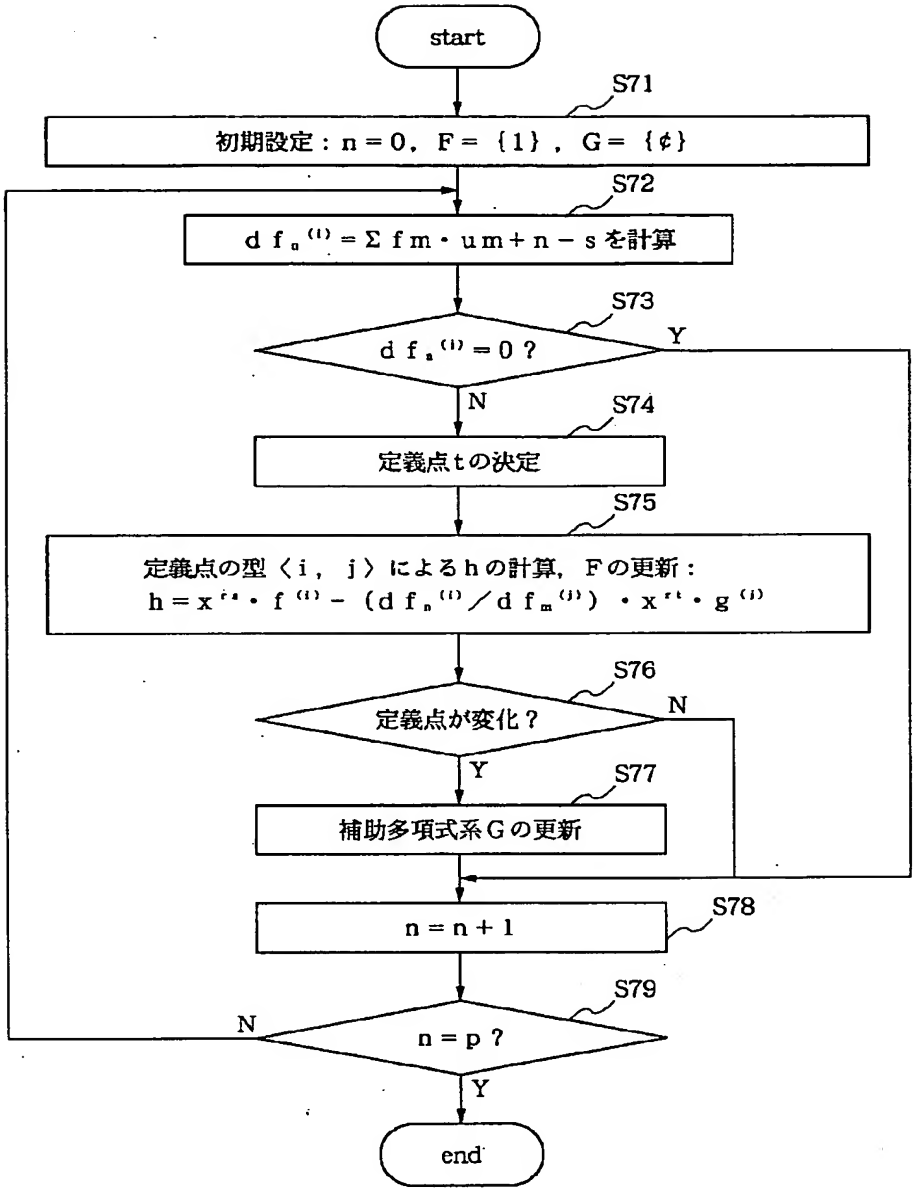
【図４】



【図６】



【図7】


start
S71
初期設定：n＝0，F＝｛1｝，G＝｛¢｝
S72
$df_n^{(j)} = \Sigma fm \cdot um+n-s$を計算
S73
$df_n^{(j)} = 0$？
Y
N
S74
定義点tの決定
S75
定義点の型〈i，j〉によるhの計算，Fの更新：
$h = x^{ri} \cdot f^{(j)} - (df_n^{(j)} / df_m^{(j)}) \cdot x^{rt} \cdot g^{(j)}$
S76
定義点が変化？
N
Y
S77
補助多項式系Gの更新
n＝n＋1
S78
S79
n＝p？
N
Y
end